EXHIBIT C

Statistical Detection of Downloaders in Freenet

Brian N. Levine, Marc Liberatore, Brian Lynn, Matthew Wright†

College of Information and Computer Sciences, University of Massachusetts Amherst, MA, USA
†Department of Computing Security, Rochester Institute of Technology, NY, USA

Abstract—Images posted to file-sharing networks without a person's permission can remain available indefinitely. When the image is sexually explicit and involves a child, the scale of this privacy violation grows tremendously worse and can have repercussions for the victim's whole life. Providing investigators with tools that can identify the perpetrators of child pornography (CP) trafficking is critical to addressing these violations. Investigators are interested in identifying these perpetrators on Freenet, which supports the anonymous publication and retrieval of data and is widely used for CP trafficking. We confirmed that 70,000 manifests posted to public forums dedicated to child sexual abuse contained tens of thousands of known CP images including infants and toddlers. About 35% of traffic on Freenet was for these specific manifests. In this paper, we propose and evaluate a novel approach for investigating these privacy violations. In particular, our approach aims to distinguish whether a neighboring peer is the actual requester of a file or just forwarding the requests for other peers. Our method requires only a single peer that passively analyzes the traffic it is sent by a neighbor. We derive a Bayesian framework that models the observer's decision for whether the neighbor is the downloader, and we show why the sum traffic from downloaders relayed by the neighbor is not a significant source of false positives. We validate our model in simulation, finding near perfect results, and we validate our approach by applying it to real CP-related manifests and actual packet data from Freenet, for which we find a false positive rate of about 2%. Given these results, we argue that our method is an effective investigative method for addressing privacy violations resulting from CP published on Freenet.

I. INTRODUCTION

Images are posted to the Internet every day without the consent of the persons captured. In cases where the images are sexually explicit, this represents a tremendous violation of privacy. Recently, popular services such as Reddit[1] and Twitter[2] changed their privacy policies to thwart this practice. This problem is many times worse when images capturing the sexual exploitation of very young children (i.e., child pornography, or CP) are posted to unmanaged forums, such as file-sharing networks. Unfortunately, CP can remain available on the Internet for many years [1], extending the privacy violation potentially throughout a victim's whole life. For example, victims report ongoing psychosis, anxiety, and other disorders decades after the abuse has ended [2], in part from knowing that the images have been viewed widely [3]. Providing investigators

with tools that can identify the perpetrators of CP file trafficking is critical to addressing these privacy violations.

In this paper, we focus on the investigation of such privacy violations on the Freenet [4] peer-to-peer system, which is itself a privacy enhancing technology supporting the anonymous publication and retrieval of data. Freenet, whose development began in 2001 [5,6], does not serve as an anonymous conduit to the Internet, unlike Tor; only previously published content can be retrieved. Documents published to Freenet are fragmented into small, encrypted *blocks* that are dispersed randomly throughout the network of peers. A *manifest key* is a URI necessary to retrieve and reconstruct the original document. Many manifest keys for files are broadcast via open, public forums, including for content that is illegal in many countries.

By searching Freenet's *freesites* and Usenet-like *Frost* system, we found a number of areas explicitly and openly dedicated to child sexual exploitation. We harvested 70,000 manifest keys from these areas and examined block requests in real Freenet traffic. Law enforcement confirmed to us that the manifests contained tens of thousands of child pornography images of which one-third were of infants and toddlers; additional previously unknown images were also present. We found that requests for documents using these manifest keys account for about 35% of Freenet traffic.

To deter these ongoing privacy violations and catch the perpetrators, investigators may be interested in finding persons who publish and request this data. Prior work has revealed some vulnerabilities in Freenet's approach [7,10,11,12,13] that investigators might try to apply to de-anonymize requesters of CP. These are relatively heavy-weight approaches that require active maneuvering of the investigator's position, the use of active probing traffic, or the use of multiple peers. Further, the Freenet developers have largely addressed these vulnerabilities.

In this paper, we design and evaluate a method for investigation of CP-based privacy violations on Freenet. Our findings also provide lessons for the design and implementation of anonymous file-sharing systems. In particular, we present a novel approach to distinguishing whether a neighboring peer is the actual requester of a document, with potential for real-world application by investigators. Our algorithm requires only a single peer and the passive analysis of the traffic that is sent to it by the neighbor. The method is based on a surprisingly simple observation. Because blocks are evenly distributed around the network, an observer who is one of g neighbors of the original downloader can expect to receive about $\frac{1}{g}$ of all requests. If the observer is actually the neighbor of a neighbor of the original downloader, then only about $(\frac{1}{g})^2$ will be received, assuming

---

the requestor has the same number of neighbors. Given the manifest key, an observing peer can determine the number of requests expected. Accordingly, given the number of requests received, the observer can quantify the probability of whether the requests were relayed by or originated with its neighbor.

Our contributions are as follows. We first derive a Bayesian framework that models the investigator's decision for whether the neighbor is the downloader. Based on this framework, we show why the traffic of downloaders that passes through the neighbor is not a significant source of false positives. We validate our model through simulation and show that our approach is very accurate. In practice, some false positives may result due to details of Freenet's operation and use that are hard to model. We thus apply our approach to real public manifests and actual packet data from Freenet, and we show that our false positive rate is about 2% in practice. Given these results, we argue that this investigative technique is a highly accurate, efficient, and effective method of addressing privacy violations resulting from CP published on Freenet. Finally, we discuss redesigns of Freenet that may allow downloaders to avoid detection.

II. BACKGROUND

We now briefly describe Freenet's overall design and operation. The original Freenet papers [5,6] provide many other details, and the Freenet wiki (https://wiki.freenetproject.org) and source code (https://github.com/freenet/fred) are also valuable resources.

Each Freenet *node* connects to other Freenet nodes, the set of which we call its *peers*. Each node contributes storage to the network. Files are inserted into the network and stored in the storage contributed by other nodes. After a file is inserted, a URI known as a *manifest key* is returned to the user. Anyone with knowledge of the manifest key can retrieve the file.

Before Freenet inserts a file into the network, it encrypts and divides the file into 32KB *blocks*. A node distributes blocks to its peers. A peer may place the block in its own storage, and it may also send the block to its own peers. Blocks are stored using a key-value pair, with the *key* being the SHA256 hash of the block, and value being the data.

Freenet has two operational modes: *opennet* and *darknet*. In opennet, nodes connect to other opennet nodes, discovered from well-known seed nodes or other peers. Freenet allows for opennet nodes to exchange peers, and form new links to peers, in order to better organize the network as a distributed hash table. In darknet, Freenet nodes connect only to peers for which the user has explicitly given permission. We consider only opennet, though our technique should work for either.

Manifests. When a file is inserted into the network, a *manifest block* is also inserted. The manifest key contains the decryption key, and the block's hash, necessary for retrieval. The manifest block contains the hashes and decryption keys of each *content block*. If the manifest block cannot reference all of the content blocks, it will reference another level of manifest blocks.

Retrieving a file is the reverse process of inserting a file. The downloading node retrieves the manifest block(s). Once the file's content blocks have been identified, they can be retrieved. Retrieving a file does not result in a file's blocks being placed into the downloading node's local storage.

If the peer receiving a request has the block in its Freenet storage, it returns the block. Otherwise, the peer forwards the request to one of its peers. This process will continue from peer to peer. When found, the block is returned in reverse order through the chain of peers and cached. If a selected peer fails to find the block, a node may forward the request to other peers before returning a result of not found.

Hops-to-Live. To prevent block requests from propagating indefinitely, Freenet uses a *hops-to-live* (HTL) counter. Generally, the HTL begins at 18 and is decremented by each relay node until it is zero, in which case a not-found error is returned.

The downloader would be obvious if Freenet were to always start with an HTL of 18. To avoid detection, the downloading node will randomly choose whether to start at 18, or to immediately decrement the HTL to 17. Once the decrement decision is made, it is permanent for all originating requests sent to that peer. Further, the same decision applies when relaying a request with an HTL of 18 to that peer; the node will only decrement the HTL if a new request would also be decremented. As a result, if a peer receives a request with an HTL of 18 or 17, its originator is ambiguous. Note that blocks are not cached by relayers if the request HTL was 18 or 17.

Data and FEC blocks. The number of blocks Freenet inserts into the network is considerably larger than the file size divided by 32KB. In addition to the data blocks, Freenet uses Reed-Solomon codes to generate forward error correction (FEC) blocks. These *check blocks* provide redundancy and the ability to recreate missing blocks. Freenet subdivides a file into *segments*. For each n data blocks in a segment, Freenet inserts $n + 1$ check blocks. Each segment cannot reference more than 256 total blocks. To recreate the data represented by a segment, Freenet must successfully fetch n blocks, using any combination of data or check blocks.

After a segment has been successfully retrieved, the node will regenerate and insert each block in the segment that it had requested but failed to fetch. To have more nodes store a file's blocks, Freenet will randomly re-insert a block that it has successfully fetched, with a 0.5% probability of being selected for re-insertion.

Peer selection and routing. A persistent *location* is randomly assigned to each opennet node. A location is a 64-bit floating point number between 0 and 1. Locations are points on a circular space, with 0 and 1 being the same point. A *distance* can be calculated between any two locations. Each block's SHA256 hash can be deterministically converted to a location; and a distance can be calculated between a node and a block.

An opennet node attempts to have the majority of its peers' locations close to its own location, with the remaining peers distributed throughout the circle. When sending a request, a node attempts to send it in the direction of the node *closest* to the block's location. Freenet performs friend of a friend routing: nodes have visibility to their immediate peers' locations, as

well as the locations of their peers' peers. All locations are considered when selecting a recipient peer. At any given instant, some peers of a node are responsible for larger parts of the location space than others. But over time as peers come and go, each peer accounts for a roughly equal fraction of the location space visible to a node. We have verified this intuitive fact in simulations of the Freenet routing protocol.

III. INVESTIGATIVE TECHNIQUE AND ANALYSIS

In this section, we discuss an investigative model for identifying whether a Freenet peer is a downloader of a given manifest. We then show that our approach is resistant to false positives potentially caused by a node relaying multiple concurrent downloaders.

A. Assumptions and Model

The goal of the investigator is to identify whether a neighbor sending requests is the *downloader* of a file in a set of files of interest, or instead relaying the requests. We assume the investigator has collected manifest keys for these files of interest, which can be obtained, for example, from Freenet message boards or web sites (i.e., *freesites*) related to CP. For simplicity, we assume the investigator operates only one peer in the Freenet network, which we call the *observer*. Running multiple peers as Sybils [14] is possible and efficiently allows for parallel investigations. The observer is strictly passive: it participates in the network by forwarding requests as usual, but also logs the SHA256 hash keys of any request it sees, together with the ID of the peer (i.e., the neighboring node) that sent it the request, and the count of the peer's neighbors. Surprisingly, these are all the steps required.

B. Description

For each manifest key from the files of interest, the investigator obtains the SHA256 hash keys for each block in the manifest, which can be retrieved from Freenet by fetching the manifest blocks. The observer node passively logs requests to download blocks from its peers, and the observer can easily map the requests to the files of interest by the key values. Requests that don't map are not logged. The observer then counts the requests received on a per-peer and per-file basis, for all known files. Based on the counts, the observer can calculate the likelihood that a given peer is either:

- requesting to download blocks for a specific file; or
- relaying requests of a third node to download blocks for a specific file.

Figure 1 illustrates the two scenarios.

Let us call the observer's peer who sends the requests the *subject*, as the subject may be the downloader or merely forwarding the requests. Because only requests with an HTL of 18 or 17 could have originated at the subject, we do not need to count requests with lower HTLs. For the remainder of this section, we consider only requests with HTLs of 18 or 17 unless otherwise noted.

The intuition for the investigative technique is now easy to describe. For large files, the downloader will make a
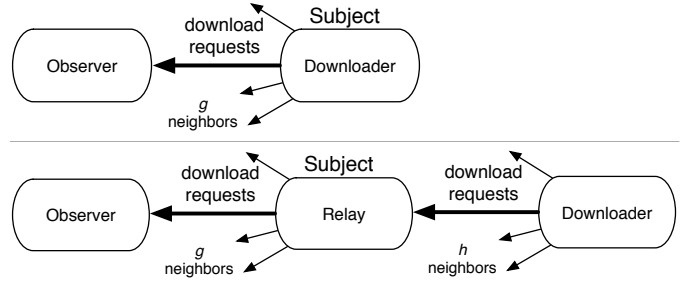

Fig. 1: The observer's goal is to distinguish between these two scenarios.

large number of requests for blocks from the manifest, and those requests will be spread randomly among its peers. An observer who is a peer of the downloader will expect to see a certain number of those requests, with some variance. On the other hand, if the observer is merely a peer to a peer of the downloader, it will see only a fraction of the requests seen by the peer. The investigative technique uses a statistical test to distinguish between these two cases.

C. Analysis

We now more formally describe and analyze the investigative technique, which uses several values as input. Two directly observed (per-file, per-peer) values are:

r : the number of download requests received by the observer from the subject;

g : the number of directly connected peers of the subject (including the observer herself).

The observer must select two additional values:

T : the total number of requests made by a downloader of the given file;

h : the number of peers assumed connected to a hypothesized source that is not the directly observed subject.

The number of requests actually observed is the upper bound of r; in practice, we use a downwardly adjusted value, as described in Section III-E. The observer learns g from the Freenet protocol. T is not simply the total number of blocks in the manifest: almost half of those blocks are redundant to ensure that the file can be downloaded even when some blocks are unavailable. Due to Freenet's implementation, the number of requests made is dependent on the number of blocks available. We define the values used for T and h in Section III-E.

We construct a model by assuming that each request the downloader makes is sent to exactly one of its peers, and that the selection of that peer is made uniformly at random. The total number of requests an observer will receive from the subject if she is the downloader can be modeled by a Binomial distribution. Let p be the probability of each request being sent to the observer. Given T possible download requests, the probability of the observer receiving r requests is

$$B(r; T, p) = \binom{T}{r} p^r (1-p)^{T-r}. \tag{1}$$

Let $H1$ be the event that the subject is the downloader, in which case $p = 1/g$. Let $H2$ be the event that a peer of the subject is the downloader, in which case $p = \frac{1}{gh}$.

As stated above, the technique assumes that either the peer is the downloader or a directly connected peer of the peer is the downloader. Using a Bayesian framework, this assumption can be modeled as follows. We seek the probability of $H1$ given that the observer has received r requests, which can be written as follows using Bayes' Rule:

$$Pr(H1|r) = \frac{Pr(H1)Pr(r|H1)}{Pr(r)} \quad (2)$$

$$= \frac{Pr(H1)Pr(r|H1)}{Pr(H1)Pr(r|H1) + Pr(H2)Pr(r|H2)}. \quad (3)$$

We know that

$$Pr(r|H1) = B\left(r; T, 1/g\right) \quad (4)$$

and similarly,

$$Pr(r|H2) = B\left(r; T, 1/gh\right). \quad (5)$$

To set the priors $Pr(H1)$ and $Pr(H2)$, we use the number of peers of the subject as a guide. Assuming that among the subject or its peers, each is equally likely to be the downloader, that is $Pr(H1) = \frac{1}{g+1}$ and $Pr(H2) = \frac{g}{g+1}$. Thus, we have

$$Pr(H1|r) = \frac{\frac{1}{g+1}B\left(r; T, 1/g\right)}{\frac{1}{g+1}B\left(r; T, 1/g\right) + \frac{g}{g+1}B\left(r; T, 1/gh\right)}. \quad (6)$$

We quantitatively evaluate this equation in Section IV.

D. Multiple Third-Party Downloaders

Beyond the two cases above, it is also possible that the subject is relaying a large number of download requests from multiple third parties, and thus could be mistaken as a downloader via Eq. 6. Below we show why this is unlikely.

If the subject is the downloader, then an observer can expect to see $\frac{1}{g}$ of the T requests with an HTL of 18 or 17. Let us consider the case that the subject is not the downloader, but it instead relays requests for multiple other nodes. In the worst case for a false positive determination for the subject, let us suppose that the set of downloaders includes *all nodes except the observer and the subject*: every peer of the subject (except the observer), as well as all of their peers, and so on. When h is small, a larger number of requests from these peers will reach the observer because, in that case, the subject will receive and potentially forward proportionally more of the requests from these peers to the observer. This holds generally: peers of the peer, and so on, in the worst case have few peers.

We can represent this set N of peers as an inward tree of nodes, constructed as follows:

- The observer has one child, the subject;
- The subject has b children, N_0^1, N_1^1, ..., N_{b-1}^1;
- The tree continues with a *branching factor* of b, such that each node in the tree has one parent and b children. Let N_j^i be the jth node at level i.

While the network structure of Freenet is not a tree, the routing of messages through the network is, to a first approximation, along a tree as described here.

Let us consider N^1, the first level of the tree. In expectation, half of these nodes will send requests to the subject with an

HTL of 17, and the subject will decrement those HTLs to 16 before forwarding them on. The investigator will discard these requests, meaning that we only need to consider requests that arrive at the subject with HTL 18. The other half of nodes in N^1 will generate requests with an HTL of 18 when sent to the subject. Accordingly, the subject will receive, in expectation, requests from $\frac{b}{2}$ of the nodes in N^1 with an HTL of 18. It will not receive all the requests generated by these nodes, as each will split requests among their $b + 1$ peers. The subject will, in expectation, receive $\frac{b}{2}\frac{1}{b+1}$ requests from the nodes in N^1 and forward $\frac{1}{b}$ fraction of these to the observer. In other words, the observer can expect to see $\frac{b}{2}\frac{1}{b+1}\frac{1}{b} = \frac{1}{2(b+1)}$ of the requests generated among the nodes at N^1.

At N^2, half of the connections to N^1 will have an HTL of 17, and any of these that eventually reach the observer would be discarded. Again, half of the connections between N^1 and the subject, where the requests arrives at N^1 with an HTL of 18 will decrement the HTL. The subject will thus receive, in expectation, requests from $\frac{b}{4}$ of the nodes in N^2 with an HTL of 18, where $\frac{1}{b+1}$ of these requests will be sent to a node in N^1. Of those, $\frac{1}{b}$ will be sent to the subject, and $\frac{1}{b}$ of those to the observer. In other words, the observer can expect to see a total of $\frac{b}{4}\frac{1}{b+1}\frac{1}{b}\frac{1}{b} = \frac{1}{4b(b+1)}$ of the requests generated from among the nodes at N^2.

In general, for a network with n levels of nodes where all nodes are requesting the same file, we expect to see a fraction of the requests equal to

$$\sum_{i=1}^{n} \frac{1}{2^i b^{i-1}(b + 1)} \quad (7)$$

if *all* nodes are downloading the file of interest.

Recall our motivating claim: that it is unlikely in practice for third-party download requests to cause the subject to mistakenly appear to be a downloader. Mathematically, we are saying it is unlikely that

$$\frac{1}{b + 1} < \sum_{i=1}^{n} \frac{1}{2^i b^{i-1}(b + 1)}. \quad (8)$$

As stated above, we can choose a conservative value of $b = 2$. For an infinitely large network where $b = 2$, then

$$\sum_{i=1}^{\infty} \frac{1}{2^i b^{i-1}(b + 1)} = \frac{2}{9}. \quad (9)$$

In other words, even if *all* nodes in the network were downloading the file *at the same time*, we would expect to see $\frac{2}{9}T$ requests for the file. This quantity is less than the fraction of requests we would expect to see if the subject were the downloader, which is $\frac{1}{b+1}T = \frac{T}{3}$. If a smaller fraction of nodes were to be requesting the file (rather than all the nodes in Freenet), say, $\frac{1}{z}$, then the number of requests we would expect to see would fall commensurately to $\frac{2}{9z}T$.

As a result of this analysis, we believe it unlikely a subject will be mistaken as a requester by Eq. 6 due to relaying multiple third party requests for a file.

E. Modifications for Real Freenet Traffic

Recall that Eq. 6 estimates the probability that a given subject is a downloader on the basis of: g, the number of peers of

the subject, which we can observe directly; r, the (adjusted) number of requests observed; h, the number of peers of a possible third-party downloader, which we estimate; and T, the number of requests the downloader made to reconstruct the data referenced by the manifest, which we estimate. In this section, we describe how to set these values given Freenet's real operation.

To apply Eq. 6 to real data, a passively observing Freenet node can log requests that are sent to it. Requests for keys, which are SHA256 values, can be compared to a table of keys harvested from manifests. Thus, any keys in the table can be mapped to specific content. Requests contain the key, an HTL, the sender's IP address and Freenet location, and the request type (retrieve or insert). The observer also logs a timestamp and the number of peers of the sender. The log is then analyzed to identify *runs* of requests. To reduce potential false positives, we define a run to be a collection of observations where:

- all observations are of data requests for blocks associated with the same manifest;
- all observations are of the same peer, as identified by IP address and Freenet location;
- all data requests have a consistent HTL: one of 17 or 18;
- a minimum of 20 data requests for distinct blocks were observed;
- and all requests were logged within a set window of time.

We use the term *data request* to be consistent with Freenet and to distinguish them from *insert requests*. A data request can be for a data block, a check block, or a manifest block. An insert request implies that a block that was requested, but not found, and is being repaired.

Recall that in Freenet's actual implementation, a manifest consists of twice as many blocks as required to recreate the original file. If all blocks are available on the network, only half would be requested. However, additional requests are made if blocks are unavailable, and the requests may be sent to multiple peers. This redundancy can inflate o, the number of distinct requests observed; and without adjustment it could lead to false positives. Therefore, for a given run, we compute

- o, the number of data requests for *distinct* blocks;
- i, the number of insert requests;
- d, the number of duplicate data requests;

and define r as

$$r = o - i - \epsilon d. \tag{10}$$

Because an insert typically indicates an additional request was issued, we decrement the count by the corresponding number. We further reduce the count by a constant multiplier, ϵ, for each duplicate key observed. Duplicate data requests represent failed data requests. We would have expected the other blocks within a segment to have been requested before re-requesting a block. Since a relayer might request a distinct block from several of its peers due to not-found errors, this can result in a large number of requests and potentially to false positives. We mitigate this by applying a multiplier to the number of duplicates observed. We used our false positive testing (see Section V-C) on real data to tune this parameter, and we determined that $\epsilon = 3$

was a reasonable trade-off between detecting downloaders and limiting false positives.

We cannot observe the total number of requests, T, made by the downloader. The minimum number of requests required to successfully download a file is approximately half the total number of blocks, due to the existence of redundant FEC blocks. To determine a suitable value for T, we conducted experiments on Freenet where we inserted our own files into the network and then instrumented a downloader to count the number of distinct blocks requested. Based on our findings, we chose a value of $T = 0.8 * TotalBlocks$.

For the value of h, the number of peers assumed to be connected to a hypothesized source, we use a value of 8, which we believe to be extremely conservative based on our observations of peer counts and reported Freenet statistics [15]. Freenet defines the minimum number of allowable peers to be 10. In other words, this very low estimate reduces the number of false positives at the cost of increasing the number of false negatives as well.

IV. EVALUATION I: SIMULATION

The goal of our evaluation is to quantitatively determine the accuracy of Eq. 6, which is the core of our investigative technique. In this section, we use a simulation that models Freenet's graph topology and basic routing mechanisms. In these simulations, we show near perfect accuracy. In Section V, we evaluate the technique on real Freenet data, resulting in a slightly higher FPR due to its more complicated mechanisms.

A. Assumptions

Freenet has a small-world topology [16], which we create in simulation via Watts and Strogatz's algorithm [17]. We assign each node a random location from $[0, 1)$. We then assign each node a set of c edges to *close* nodes, and also a set of l *long-distance* edges, where distance is defined using the Freenet distance metric. The c edges to close nodes are chosen uniformly at random from among the $2c$ closest nodes, and the l edges are chosen uniformly at random from among the remaining nodes. In the end, each node has *at least* $c + l$ edges due to edge selection.

All our graphs were constructed with 5,000 nodes total. In any single trial, all nodes in a graph had the parameters $c + l$ of either 27+3, 54+6, 81+9, or 108+12, resulting in average degrees of 36, 72, 108, and 144, respectively. For each degree, we constructed 500 random graphs. The real Freenet graph is comprised of nodes with a variety of degrees, at or below these values [18].

For each graph, we requested blocks from a random node to one of the 5,000 locations in the graph. To find the content, we use the Freenet friend-of-a-friend routing algorithm, HTL decrementing, and other critical details (see Section 2).

Specifically, we ran two types of trials on 2,000 graphs constructed using the four different degrees. In the first type of trial for a specific graph, 5,000 times we selected a node at random as an originator, another at random as a destination, and then one of the originator's *adjacent* neighbors at random
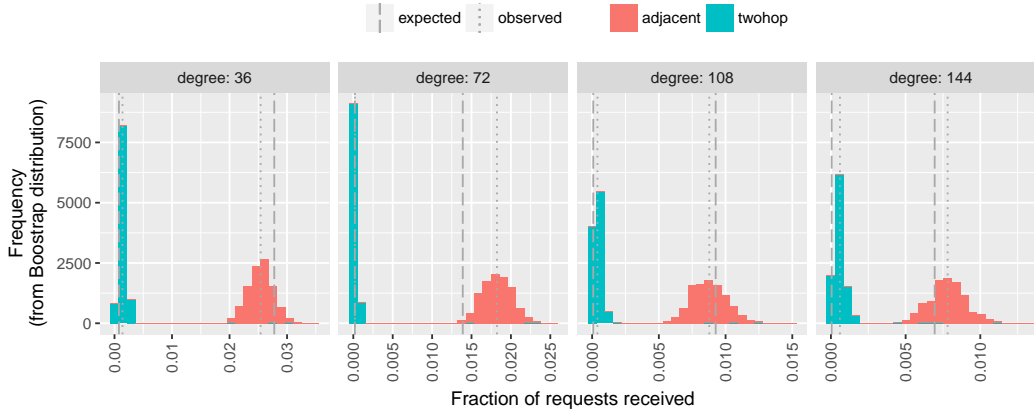
Fig. 2: The distribution of requests received by an observer either adjacent to or two hops away from the requester. The distribution for each type of observer is based on 10,000 bootstrapped samples from one example graph with node degrees of 36 to 144. Even visually, it's clear observers can differentiate the two scenarios. See Figure 3 for full results using 500 graphs per degree.
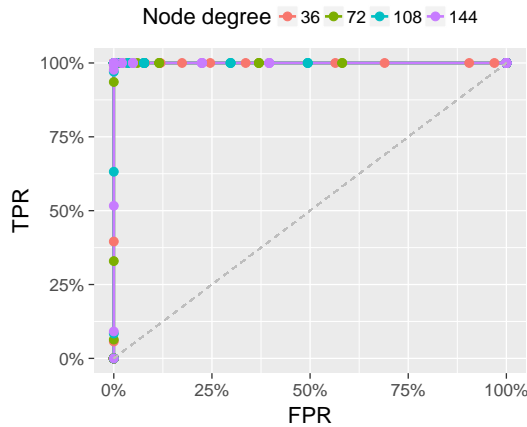

Fig. 3: ROC curve for investigator success based on Eq. 6, from trials for 500 random small-world graphs for each degree (i.e., 2,000 graphs total).

as an observer. In the second trial, 5,000 times we selected a node at random as an originator, another at random as a destination, then one of the originator's neighbors *two hops* away at random as an observer. Each of the 5,000 requests resulted in a path that did or did not include the observer. Next, for each trial, we created 10,000 *bootstrapped samples* [19], where each sample was constructed by selecting 5,000 paths randomly with replacement from that trial's 5,000 requests.

B. Results

Figure 2 provides an intuitive explanation of why requestors are distinguishable from neighbors that are relaying requests in Freenet. Each facet of the figure shows the results for the two primary trials on exactly one sample graph with a specific degree. Each facet plots the fraction of requests observed from the 10,000 bootstrapped samples for an example trial. One distribution of bootstrapped samples is presented for the case when observer is adjacent to the requester, and another distribution for when it is two hops away. A simple visual test distinguishes the two scenarios. For these experiments, observations included only packets with HTLs of 18 or 17.

In each plot, vertical lines shows the observed mean fraction of requests for the two scenarios, and two other lines show a simple expected fraction of 1/degree for the adjacent and

$(1/\text{degree})^2$ for the two-hop case. Because nodes in the graph have different degrees, and requests are sent from a random node, there is not an exact match with the mean degree.

Figure 3 shows a Receiver Operating Characteristic (ROC) curve for the result of applying Eq. 6 to all 10,000 bootstrapped samples for each of the two trials for each of the 2,000 graphs. The plot shows FPR versus TPR as a curve parameterized by a threshold from 0 to 1 for the value of Eq. 6. A true positive is an observation from an adjacent node with a probability greater than or equal to the threshold. A false positive is an observation from a two-hop neighbor with the same.

As the figure shows, the algorithm obtains near-perfect accuracy. The area under the curve is numerically equivalent to 1.0 for all four graph types. The equal error rates (i.e., where $\text{FPR} = 1 - \text{TPR}$) for each graph is as follows: 0.0 for degree of 36; 0.0000002 for degree 72; 0.000012 for degree 108; and 0.00034 for degree 144. The high accuracy is due to the distinct separation of the distributions, illustrated in Figure 2.

V. EVALUATION II: REAL FREENET DATA

In this section, we quantitatively evaluate our investigative technique by applying it to real Freenet CP-related manifests and requests. We find that our approach has a low false positive rate of about 2% for these data. We also discuss challenges in redesigning Freenet to thwart downloader detection.

A. Child Exploitation on Freenet

We harvested more than 70,000 manifest keys posted to freesites and Frost boards explicitly dedicated to child exploitation. We queried Freenet for the block keys associated with those manifests, which resulted in the collection of over 150 million distinct keys. We did not download the files.

Many files inserted into Freenet are *zip* or *rar* archives. But for manifests containing non-archived files, we identified the SHA hashes of 54,000 distinct non-archived images using file names and meta data from manifest blocks (some images appeared in multiple manifests). We provided the hashes to law enforcement, and 31,000 were known to them as child sexual exploitation material, of which 9,000 were flagged as having infants or toddlers as victims. Law enforcement confirmed from spot downloads that some of the unknown files and archive-based manifests included previously unknown CP.

These figures are in line with previous studies. CP is not "sexting" crimes by late teens: Wolak found that 21% of CP possessors have images depicting sexual violence to children such as bondage, rape, and torture; 28% have images of children younger than 3 years old [20]. Studies have shown that at the time of arrest 10%–12% of possessors are found to be also physically abusing children [1,21]; post-arrest, it increases to 58%–85% from, for example, self-admission during counseling or victims coming forward [22,23]. And CP sharing on peer-to-peer networks is vast. We measured over 600,000 unique peers per month in 2015 sharing known CP in file sharing networks such as BitTorrent [1]. For this paper, we re-measured that statistic and found 1,200,000 unique peers in February 2017.

We condemn the misuse of the techniques of this paper to investigate ethical uses of Freenet.

B. Experimental Setup

We ran passive Freenet opennet nodes from November 2016 through January 2017, inclusive. Our nodes were modified to log only the requests whose keys matched those we harvested. Of all the block requests observed with an HTL of 17 or 18, 35% were for those keys that we had identified from forums and sites associated with child exploitation. As noted above, law enforcement identified over 31,000 known CP images in these manifests, and though some remain unconfirmed, it is also unlikely that we located all CP manifests in Freenet. *In sum, it's reasonable to assume that at least about 35% of Freenet's traffic is related to child exploitation material.*

The most recent data from [15], which stopped reporting Freenet statistics in September 2016, estimated that the size of the Freenet network averaged 8,000 active nodes. Over the duration of the last month of our measurements, January 2017, our nodes were peered randomly with over 42,000 distinct Freenet nodes, and about 4,200 distinct nodes per day. Tor is reported to have many more users [24], and so is BitTorrent [1]. However, since the trafficking of child exploitation materials is illegal in many countries, and because Freenet is heavily used for trafficking CEM, there is motivation for law enforcement to focus investigations on Freenet.

C. False Positive Rate on Real Data

Because of Freenet's policy for decrementing HTLs and its defined maximum HTL of 18, we can assume that requests with an HTL of 16 or below did not originate with neighbors. Under this assumption, any positives our algorithm would report on requests with HTLs of 16 are false positives. (We cannot calculate a TPR for these manifests.)

Using six weeks of data — from the first two weeks of November 2016, December 2016, and January 2017 — we identified 26,963 *runs*, as defined in Section III-E except replacing the requirement that the HTL be 17 or 18 with a requirement that the HTL be 16. All told, $323/26963 = 1.2\%$ runs tested positive, if we set a threshold probability of 98% or higher using Eq. 6. We note that there were nearly zero false positives in our simulations; false positives in real Freenet are

likely due to, for example, duplicate data requests, which we can only partially account for using the selected value of ϵ.

A limitation of our findings is that HTLs of 16 are, on average, more hops from the downloader than relayers with 17s and 18s. However, we can adjust the FPR rate as follows. The degree distribution of neighbors varied; past work has observed that Freenet nodes vary typically from the default minimum of 10 to about 100 [18]. Using our simulation, we know empirically that, for a network comprised of only nodes with degree 10, the chances a request with an HTL of 16 is 2 hops away is 52%. Therefore, we adjust our FPR by assuming the worst case: that in our 26,963 runs, only 52% (14,021) runs were from neighbors 2 hops away; and furthermore all 323 runs testing positive were from that set. In other words, we estimate the FPR for our test of real data could be as high as $323/14021 = 2.3\%$. (N.b., the adjusted FPR would be lower if we assumed the node degree was higher, which it surely is.)

D. Avoiding Detection

A significant redesign of Freenet is required to prevent detection of downloaders. Below we detail why simple changes are insufficient. Thus, proposing and evaluating a new design for Freenet is beyond the scope of this preliminary work.

Sometimes, never, or always decrement. Currently, Freenet peers decrement the HTL of requests with an HTL of 18 with probability $p = 0.5$; this coin flip is performed once per edge. A downloader that flips per-packet or per-manifest advantages the investigator. Another approach is to alter p, but any $p \neq 0.5$ allows for very effective downloader detection. When $p < 0.5$, an HTL of 18 will occur more frequently, revealing a downloader; when $p > 0.5$ an HTL of 17 will similarly occur more frequently, revealing a downloader. Setting p to 0 or 1 is the investigator's best case.

Choose a different initial HTL. Our investigative technique does not consider requests with HTLs of 16 or lower because they are sent by only relays. Downloaders could choose a different initial (maximum) HTL, resulting in values other than 17 or 18 being sent to their neighbors. This is not an effective approach; it requires only that the investigator log additional HTLs. As long as the initial HTL is selected once per edge, the investigator can still construct runs from only a single HTL at a time. Further, we expect the investigator can use statistical inference to determine the initial HTL selected by the downloader.

Per-packet decrement. Another approach is for the downloader to decrement HTLs on a per-packet basis by some integer $d \geq 0$, requiring the investigator to include requests with multiple HTLs in each run, which might increase the FPR. If we assume the investigator wishes to minimize her FPR, this approach would force the investigator to use a threshold that drives down the TPR as well. But the approach is not a clear win. First, we expect the investigator could statistically infer the downloader, given the HTLs for each request associated with a particular manifest. On average, higher HTLs would be expected from the originator. It's non-obvious what algorithm selects d in a way that prevents this

inference. Second, imagine that such an algorithm exists; in that case, including multiple HTLs in runs could increase the FPR. However, our basic approach would still apply and would remain effective, especially for manifests that are less popular. Quantifying whether the FPR increase is significant would require an update of our analysis in Section III to include manifest popularity. We leave this analysis for future work.

Removing HTLs. Freenet could be redesigned completely to remove HTLs from requests, which is the method employed by OneSwarm [25]; however, this approach is not secure [26,27].

VI. RELATED WORK

In comparison to past work on Freenet vulnerabilities, ours is the only passive method, and ours is the only method that requires but a single peer.

Baumeister et al. [11] discovered a Routing Table Insertion (RTI) attack that does not de-anonymize a peer, but it can be used by other methods to traverse the network. This vulnerability can be addressed with randomized routing [28]

Tian et al. [7,10,12] discovered a Traceback Attack in Freenet that exploits a unique identifier (UID) assigned to each request as confirmation that the peer must have been on the path from the original requester. By actively probing all neighbors of the peer, and leveraging the RTI attack, the attacker can move toward the requestor. Freenet developers addressed the attack by having peers discard a UID after receiving the response to the outstanding request; see also [8,9].

Roos et al. [13] show how Freenet network probes, intended to gather obfuscated values, can be used to infer the actual value with a Bayesian model after multiple observations. The attack is a general approach, but a specific example to infer bandwidth is provided, which could be used to detect opennet-darknet bridges; see also [18].

OneSwarm [25] is an anonymous filesharing network with similarities to Freenet. OneSwarm does not use an HTL field; instead a cancel message chases down the flooded request with purposeful delays. Bissias et al. [26,27] demonstrated that this approach cannot hide the source of messages, suggesting there is no method by which Freenet can drop its HTL altogether.

VII. CONCLUSIONS

We presented a passive technique for detecting Freenet downloaders who violate the privacy of sexually abused children. We obtained 70,000 manifests posted to forums openly dedicated to child sexual exploitation and confirmed to include known CP images; we found that about 35% Freenet's traffic is related to these manifests. Our approach requires only a single peer and known child pornography manifest keys. We derived a Bayesian framework for testing whether a peer may be downloading a document, based on counting the requests observed. We have demonstrated that the investigative technique is resistant to false positives when requests from multiple nodes are being relayed by a single peer. We have validated our model through simulation. We demonstrated an FPR on actual Freenet traffic of approximately 2%.

REFERENCES

[1] G. Bissias, B. N. Levine, M. Liberatore, B. Lynn, J. Moore, H. Wallach, and J. Wolak, "Characterization of Contact Offenders and Child Exploitation Material Trafficking on Five Peer-to-Peer Networks," *Child Abuse & Neglect*, 52:185–199, Feb 2016.

[2] M. Cutajar, P. Mullen, J. Ogloff, S. Thomas, D. Wells, and J. Spataro, "Psychopathology in a large cohort of sexually abused children followed up to 43 years," *Child Abuse & Neglect*, 34(11):813–822, 2010.

[3] E. Bazelon, "The price of a stolen childhood," *New York Times Magazine*, https://nyti.ms/2kmwJlJ, Jan 27 2013.

[4] Freenet reference daemon source code, https://github.com/freenet/fred.

[5] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Proc. Intl. Wkshp. Designing Privacy Enhancing Technologies*, 2001, pp. 46–66.

[6] I. Clarke, S. Miller, T. Hong, O. Sandberg, and B. Wiley, "Protecting free expression online with Freenet," *IEEE Internet Computing*, 6, no. 1, pp. 40–49, Jan 2002.

[7] G. Tian, Z. Duan, T. Baumeister, and Y. Dong, "A traceback attack on Freenet," in *Proc. IEEE INFOCOM*, Apr 2013, pp. 1797–1805.

[8] ——, "Thwarting traceback attack on freenet," in *Proc. IEEE GLOBECOM*, Dec 2013, pp. 741–746.

[9] ——, "Reroute on loop in anonymous peer-to-peer content sharing networks," in *Proc. IEEE Conf. Communications and Network Security*, Oct 2014, pp. 409–417.

[10] ——, "A traceback attack on Freenet," *IEEE Trans. Dependable Secure Comput.*, no. 10.1109/TDSC.2015.2453983, Jul 2015.

[11] T. Baumeister, Y. Dong, Z. Duan, and G. Tian, "A Routing Table Insertion (RTI) Attack on Freenet," in *Proc. Intl. Conf. on Cyber Security*, 2012.

[12] R. Rajan, "Feasibility, Effectiveness, Performance and Potential Solutions on Distributed Content Sharing System [plagiarized]," *Intl. J. Engineering and Computer Science*, 5(1):15638–15649, Jan 2016 http://www.ijecs.in/issue/v5-i1/30%20ijecs.pdf.

[13] S. Roos, F. Platzer, J. Heller, and T. Strufe, "Inferring obfuscated values in freenet," in *Proc. NetSys*, Mar 2015, pp. 1–8.

[14] J. Douceur, "The Sybil Attack," in *Proc. Intl. Wkshp. Peer-to-Peer Systems*, Mar. 2002, pp. 251–260.

[15] S. Dougherty, "Freenet statistics," https://www.asksteved.com/stats/.

[16] H. Zhang, A. Goel, and R. Govindan, "Using the small-world model to improve freenet performance," in *Proc. Infocom*, 2002, pp. 1228–1237.

[17] D. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, 393(6684):440–442, 1998.

[18] S. Roos, B. Schiller, S. Hacker, and T. Strufe, "Measuring freenet in the wild: Censorship-resilience under observation," *Proc. Privacy Enhancing Technology Symposium*, LNCS 8555, pp. 263–282, Jul 2014.

[19] B. Efron, "Bootstrap methods: another look at the jackknife," in *Breakthroughs in Statistics*. Springer, 1992, pp. 569–593.

[20] J. Wolak, D. Finkelhor, and K. Mitchell, "Trends in Arrests of "Online Predators"," UNH Crimes Against Children Research Center, http://www.unh.edu/ccrc/pdf/CV194.pdf, Tech. Rep., March 2009.

[21] M. Seto, R. Hanson, and K. Babchishin, "Contact sexual offending by men with online sexual offenses," *Sex Abuse*, 23(1):124–145, 2011.

[22] M. Bourke and A. Hernandez, "The butner study redux: A report of the incidence of hands-on child victimization by child pornography offenders." *Journal of Family Violence*, 24(5):183–191, 2009.

[23] M. Bourke, L. Fragomeli, P. Detar, M. Sullivan, E. Meyle, and M. O'Riordan, "The use of tactical polygraph with sex offenders," *Journal of Sexual Aggression*, 21(3):354–367, 2014.

[24] Tor metrics, https://metrics.torproject.org/userstats-relay-country.html.

[25] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson, "Privacy-preserving P2P data sharing with OneSwarm," in *Proc. ACM SIGCOMM*, Aug 2010, pp. 111–122.

[26] G. Bissias, B. N. Levine, M. Liberatore, and S. Prusty, "Forensic Identification of Anonymous Sources in OneSwarm," *IEEE Trans. Dependable Secure Comput.*, no. 10.1109/TDSC.2015.2497706, 2015.

[27] S. Prusty, B. N. Levine, and M. Liberatore, "Forensic Investigation of the OneSwarm Anonymous Filesharing System," in *Proc. ACM CCS*, Oct 2011, pp. 201–214.

[28] T. Baumeister, Y. Dong, G. Tian, and Z. Duan, "Using randomized routing to counter routing table insertion attack on freenet," in *Proc. IEEE GLOBECOM*, Dec 2013, pp. 754–759.